For a much more detailed processing guide, please see the PASSCAL document: "Generating SEED From RT130 Data", available from our web page: http://www.passcal.nmt.edu/content/data-archiving

```
#comment: This is a batch file example.

net PI Pier database at PASSCAL

sta ME42 -77.72237 162.27354 0.042 Socorro, NM, USA
time 02/06/2011 00:00:00
datalogger  rt130  0984
sensor cmg3t 0 T4476
axis Z 0 0 - 1 1
axis N 0 90 - 2 1
axis E 90 90 - 3 1
samplerate 40sps
channel Z BHZ
channel N BHN
channel E BHE
samplerate 1sps
channel Z LHZ
channel N LHN
channel E LHE
add

close ME42 12/31/2013 23:59:59

sta ME101 -77.72591 162.26907 0.079 Socorro, NM, USA
time 02/29/2011 00:00:00
datalogger  rt130  0988
sensor cmg3t 0 T4723
axis Z 0 0 - 1 1
axis N 0 90 - 2 1
axis E 90 90 - 3 1
samplerate 40sps
channel Z BHZ
channel N BHN
channel E BHE
samplerate 1sps
channel Z LHZ
channel N LHN
channel E LHE
add

close ME101 12/31/2013 23:59:59
```

Last revised: January 16, 2012

RT130 Data Processing In a Nutshell

You've offloaded a service run and have stacks of zip files. Now what to do with them? Start by getting organized (steps 1-4). Then convert the data & log files to miniseed (5-8). Build the database (9-13) and the dataless (14-15). Finally, send the day volumes and the dataless to PASSCAL (16). Turquoise highlighting indicates a Unix command line entry.

1. Create and maintain an organized directory structure for your data. For example you might start with: mkdir raw (move your raw files here) and mkdir logs (to eventually put your logs in).

2. Create a batch file describing every station in your network; use your preferred text editor and name it whatever you like. See the template on page 4 to get started. Be very accurate with your entries – small typos now can cause big headaches later.

```
#das; refchan; refstrm; netcode; station; channel; samplerate; gain
9306;   1;        1;     PI;     ME42;   BHZ;    40;          1
9306;   2;        1;     PI;     ME42;   BHN;    40;          1
9306;   3;        1;     PI;     ME42;   BHE;    40;          1
9306;   1;        9;     PI;     ME42;   VM1;    0.1;         1
9306;   2;        9;     PI;     ME42;   VM2;    0.1;         1
9306;   3;        9;     PI;     ME42;   VM3;    0.1;         1
9306;   1;        2;     PI;     ME42;   LHZ;    1;           1
9306;   2;        2;     PI;     ME42;   LHN;    1;           1
9306;   3;        2;     PI;     ME42;   LHE;    1;           1
```

3. Convert the batch file to a parameter file that our tool, rt2ms, can parse. Use batch2par to make the conversion: batch2par batch_file > par_file. The format of the final parameter file should look similar to the one above. Notice this example describes three data streams for one site only – 40sps (refstrm 1), 0.1sps (refstrm 9, the mass positions), and 1 sps (refstrm 2). **Please note** that the 'refstrm' column will need to be hand-edited because the initial output will not be understood by rt2ms. You could also make the parameter file manually, using the format shown above and the information in the batch file.

4. Make a list file showing the path and file names to be converted by rt2ms. Our example file (box on right) is called list.file

```
../raw/2011257.9E02.ZIP
../raw/2011260.9E07.ZIP
../raw/2011264.9E0F.ZIP
../raw/2011285.9E00.ZIP
```

5. Convert the raw RT130 data to miniseed.
First try rt2ms –h to see all the options you may wish to include. The actual command line should look something like this:
rt2ms -F list.file –R –Y –L –o mseed -p par_file >& rt2ms.out This process can take a while, especially for big data sets. After rt2ms finishes, move all of your .log and .err files to your logs directory.

6. Now that you have viewable files, use pql or pqlx to view the trace files and logpeek for the logs. Obvious signs of trouble: loss of GPS timing, overlaps, gaps, corrupted files, etc. Make a note of any problems. Use fixhdr to confirm the conversion is complete and to correct any problem headers, mark timing issues, and/or to convert the files to big endianess if they aren't already.

7. Copy the local log2miniseed parameter file into your working directory by typing: cp $ANTELOPE/data/pf/log2miniseed.pf . Using any text editor, change the default string in the log2minseed.pf file from this:
wfname %Y/%j/%{sta}.%{chan}.%Y:%j
to: wfname day_volumes/%{sta}/%{sta}.%{net}.%{loc}.%{chan}.%Y.%j
This long string specifies an organized directory and filename structure, as required by PASSCAL.

8. Convert the log files to miniseed format. You want to be sure the log2miniseed command is calling the parameter file you just revised:
For tsch, type: setenv PFPATH $ANTELOPE/data/pf:.
For bash, type: export PFPATH=log2miniseed.pf
Then: log2miniseed –n PI –s AB00 logs/2005:128:15:09.0965D.log
where PI, AB00, and logs/*.log are changed to your network, station, path and filenames. Do this for every log file or write a script for it.

9. Build the Antelope database by giving it a nickname and calling the batch file you made in step 2: dbbuild -b dbname batch_file >& dbbuild.out

10. View your database: dbe dbname. Scroll through a few tables to be sure it's healthy. If you find errors or inaccuracies, correct the batch file and repeat steps 9 & 10. At this point you have only a descriptive framework - the trace files are attached next.

11. When you're reasonably certain the database is error-free, link in the waveforms: miniseed2days -d dbname -U -w "day_volumes/%{sta}/%{sta}.%{net}.%{loc}.%{chan}.%Y.%j" mseed/ >& msd2days.out
This line specifies an organized directory path and the required filename structure. This command also packs the files into station-channel-day-volumes.

12. Assign calibration values from the calibration table to the wfdisc table: dbfix_calib dbname

13. Verify the correlation of your data and database:
dbversdwf –tu dbname (a good result is: 0 bad files /0 bad records*).* Also run: dbverify –tj dbname >& dbverify.out. Check the resulting dbverify.out file for errors. If necessary, fix the batch file and repeat step 9.

14. Create the dataless SEED volume with the following naming convention:
mk_dataless_seed –v –o NN.YY.dbname.YYYYDOYHHMM.dataless dbname
Where: **NN** is your network code, **YY** is the year of your data, and **DOYHHMM** is the approximate **current** day-of-year-hour-minute.

15. Verify the dataless:seed2db –v NN.YY.dbname.YYYYDOYHHMM.dataless
Note that the dataless is the final manifestation of your meta-data. It's like a menu or an index that you and future users can use to see what data are available. If any station or time range is missing from the dataless, the corresponding data are orphaned and totally inaccessible by anyone. If this should happen, find and correct the error in the batch file, then make another database and create a new dataless.

16. Last step: Please drop a note to *data_group@passcal.nmt.edu* before sending the data to PASSCAL, so that we can set up a receiving area. We prefer you use our tool gui_DoFTP. It will pack and send your files automatically, trigger the automated system on our end to begin processing, and save us all a little time and trouble.

A few tips…

Many database errors can be avoided by rounding the start & close times in the batch to 00:00:00 and 23:59:59, respectively. It's better to start early and close late (even by a day or two) to be certain that all data & log files are described in the dataless.

Station changes, such as a new datalogger, sensor or sample rate, **must** be documented in the batch file as shown in: man dbbuild_examples.